L8: Entry 23 of 26 File: USPT Mar 13, 1990

DOCUMENT-IDENTIFIER: US 4908612 A
TITLE: Computer input-output device

Detailed Description Text (36):
A KLISP program design can be visualized as a set of upside down trees each having a plurality of branches with each branch usually splitting into further branches. Each branch is made up of a list of commands called a procedure. As shown in the example of FIG. 8 each procedure is headed by a PRC instruction. The argument of the PRC instruction specifies the procedure number. Following the PRC instruction in a procedure is a label command identified by the source code abbreviation LBL. The argument of the LBL command specifies the names of the procedures. Execution of a procedure by the CPU 65 continues until another label command is reached. A label command always stops the execution of a procedure. When a label command is encountered, the name in the argument is displayed in one of the sections of the key label displays 33 and 35 as the name of an option available to the user and the KLISP program execution is halted. If the user selects a procedure by depressing the selection key 37 or 39 associated with the section of the display 33 or 35 displaying the procedure name, all of the commands in the procedure are executed by the CPU 65 in the order in which they are listed until a next label command is encountered.

upside down trees

L8: Entry 21 of 26 File: USPT Aug 14, 1990

DOCUMENT-IDENTIFIER: US 4949243 A
TITLE: Data processing system intended for the execution of programs in the form of search trees, so-called or parallel execution

Brief Summary Text (2):
In recent years several different programming languages and programming structures have been developed, at which the execution order of the programs can be described in the form of a search tree. One example of such is the programming language Prolog (Programming in Logic). Such programs are characterized in that a computer successively searches through an upside-down tree commencing at the root of the tree. For each branch and, respectively, at the branching points of the branch the computer chooses to check one of the branches at each branching point, whereafter it reverses and checks the other branch at the branching point, a.s.o. (and so on). At the execution of such a program, thus, many different branches are to be checked. At a branching point two computers can be allowed to operate in parallel, each of the computers checking one branch. This presupposes, however, that both of the computers comprise memories, which are updated with the information defining the branching point and the data being relevant in the branching point, i.e. so-called environment actual in the branching point. It is obvious, that the information including data constituting the so-called environment of the branching or division point can be very comprehensive.

L30: Entry 15 of 36 File: USPT Oct 6, 1998

DOCUMENT-IDENTIFIER: US 5819094 A
TITLE: Apparatus for log data collection and analysis

Detailed Description Text (25):
Before starting the apparatus of the present invention, an executable computer program, or an object program, should be prepared by compiling a source program written in a source language. With compiler options, the object program may contain some special instructions for calling log collection functions that perform logging of the program operations. Those call instructions are inserted by the compiler at the beginning and the end of the program body, and at the entrance and exit of each library function when it is called in the program. Further, when compiling the program, the compiler creates a cross-reference table that associates each function with corresponding line numbers in the source program and their respective addresses in the object program.

Detailed Description Text (44):
[S4] Address information is extracted from the record found in step S3. This address information resides in the caller address information field 23 explained in FIG. 2. The apparatus obtains a corresponding function name and line number by searching a cross-reference table using the address information as the keyword. This cross-reference table has been created previously by the compiler so that it will associate each function name with corresponding addresses as well as with program line numbers. Because addresses collected in this cross-reference table are discrete ones, the address extracted from the log record may not perfectly coincide with the table entries. Thus the logged address should be examined whether it is within a range between any two consecutive caller addresses, and a function name and line number corresponding to the nearest address will be finally chosen by the apparatus.

Detailed Description Text (48):
FIG. 6 shows an example of the history diagram generated through the above-described process in response to the user's request for focusing on the line number "3" in the function "TEST." The display position analyzer first obtains an address corresponding to the entered line number by consulting the cross-reference table. Then it searches the log file for records matching with the obtained address. As a result of this search-by-address operation, the relevant segments of the history diagram are displayed distinguishably from the others. That is, the segments "b" to "d", indicative of library functions called at the line number "3" in the source program, are displayed with a higher video intensity, while the rest of the history diagram is given a low intensity.

Detailed Description Text (52):
[S12] The display position analyzer obtains an address corresponding to the designated function name and line number by consulting the cross-reference table.

L30: Entry 31 of 36 File: USPT Aug 28, 1984

DOCUMENT-IDENTIFIER: US 4468732 A
TITLE: Automated logical file design system with reduced data base redundancy

Brief Summary Text (20):
To aid in the description of the preferred embodiments, a PL/1 code listing of the associative file design program method is presented as Appendix A at the end of the specification. Throughout the description provided in the specification, cross-reference is made to code identified by line numbers in the listing.

L48: Entry 6 of 10 File: USPT Aug 31, 1999

DOCUMENT-IDENTIFIER: US 5946482 A
TITLE: Method and apparatus for using parameters to simulate an electronic circuit

Detailed Description Text (39):
The computation burden of traditional "SPICE" simulation can also be reduced for a large group of passive elements, by replacing the large group of passive elements with a single "SPICE" node that contains a closed-form description of a sub-circuit being modeled. In this implementation, code is added to a "SPICE"-based simulator which computes the sub-circuit's frequency behavior contemporaneous with traditional "SPICE" analysis.

L48: Entry 9 of 10 File: USPT Apr 14, 1998

DOCUMENT-IDENTIFIER: US 5740421 A
TITLE: Associative search method for heterogeneous databases with an integration mechanism configured to combine schema-free data models such as a hyperbase

Detailed Description Text (57):
The following method is used for normalization of every group of nodes that has the same label associated with each node in the group. The group of nodes is replaced by a single node that has: (i) a list of children consisting of the concatenation of all the children of all the nodes in the group, and (ii) a set of parents consisting of the union of all the parents of all the nodes in the group. In addition, the new node is a reference to all the nodes that were referenced by any of the nodes in the group. The process is illustrated in FIG. 12(a), where nodes 1201 and 1203, denoted by n.sub.1 and n.sub.2, respectively, each have the same label denoted by 1. According to the process, nodes n.sub.1 and n.sub.2 are replaced by node 1209, denoted by n in FIG. 12(b).

Detailed Description Text (58):
The method that is used for normalizing every group of nodes, each of which is referenced by the same reference node, is similar to the above method for nodes with the same label--the group of nodes is replaced by a single node that has: (i) a list of children consisting of the concatenation of all the children of all the nodes in the group; and (ii) a set of parents consisting of the union of all the parents of all the nodes in the group. In addition, the new node is a reference to all the nodes that were referenced by any of the nodes in the group. The process is illustrated in FIG. 13(a), where nodes 1301 and 1303, denoted by n.sub.1 and n.sub.2, respectively, each have the same reference node 1305. According to the process, nodes n.sub.1 and n.sub.2 are replaced by node 1309, denoted by n in FIG. 13(b).

Detailed Description Text (59):
The method for normalizing every group of nodes, each node of which has no label, is not a referred node, and has the same list of children, comprises: replacing the group of nodes by a single node that has: (i) a list of children consisting of the children of the nodes in the group, and (ii) a set of parents consisting of the logical union of all the parents of all the nodes in the group. In addition, the new node is the reference node to all the nodes that were referenced by any of the nodes in the group. The process is illustrated in FIG. 14(a), where nodes 1401 and 1403, denoted by n.sub.1 and n.sub.2, respectively, each have the same children, nodes 1405 and 1407, denoted by n5 and n6, respectively. According to the process, nodes n.sub.1 and n.sub.2 are replaced by node 1409, denoted by n in FIG. 14(b).

Detailed Description Text (60):
The method for normalizing every group of labels which are the same is to replace the labels by a single label, and associate that label with all the nodes that were associated by the labels in the group. The process is illustrated in FIG. 15(a), where nodes 1501 and 1503, denoted by n.sub.1 and n.sub.2, respectively, each have the a label 1. According to the process, the labels are replaced by a single label that is associated with both node n as illustrated in FIG. 15(b).